Amendments to the Specification:

Replace paragraphs [0003], [0020]–[0022], [0026]–[0030], [0032], and [0037] in their entireties with the paragraphs shown below.

[0003] FIG. 1 illustrates a prior art sequencing system 100. The system 100 includes a first arbiter 102, a second arbiter 102 and a third arbiter 103 and multiple buffers 104, 106, 108 and 110. In a typical embodiment, the buffers are first in and first out (FIFO) buffers. Each of the buffers 104-110 include multiple command threads, such as 112, 114, 116, 118 stored therein. Moreover, the system 100 is divided into resource divisions, such as an ALU resource division 120 and a texture fetch resource division 122. In the ALU resource division 120, the command thread 118 may be received from an input command ~~120~~ 124 as selected by the arbiter ~~106~~ 101. The command thread 118 may then be withdrawn from the reservation stations 104 and 108 for the purpose of being provided to an ALU (not shown) and the command threads within the texture fetch resource 122 maybe withdrawn from the reservation stations 106 and 110 to be provided to a texture fetch processors (not shown).

[0020] ALU arbitration proceeds in the same way as fetch arbitration. The ALU arbitration logic chooses one of the pending ALU clauses to be executed. The arbiter selects the command thread by looking at the reservation stations, herein vertex and pixel reservation stations, and picking the first command thread ready to execute. In one embodiment, there are two ALU arbiters, one for the even clocks and one for the odd clocks. For example, a sequence of two interleaved ALU clauses may resemble the following ~~sequence, (E and~~ sequence: (E and O stands for Even and Odd sets of 4 clocks) Einst0 Oinst0 Einst1 Oinst1 Einst2 Oinst2 Einst0 Oinst3 Einst1 Oinst4 Einst2 Oinst0. As such, this way hides the latency of 8 clocks of the ALUs. Moreover, the interleaving also occurs across clause boundaries, as discussed in greater detail below.

[0021] FIG. 4 illustrates another embodiment of a multi-thread command processing system 300 having a first reservation station 302, a second reservation station 304, an arbiter 306, an ALU 308 and a graphics processing engine 310. In this embodiment, the first reservation station 302 is a pixel reservation station such that the command threads 312, 314 and 316 contain pixel-based commands therein. Furthermore, in this embodiment the second reservation station 304 is a vertex reservation station is directed towards vertex command threads illustrated as command threads 318, 320 and 322.

[0022] ~~Not~~ Although not illustrated in FIG. 4, in one embodiment an input arbiter provides the command threads to each of the first reservation station 302 and the second reservation station 304 based on whether the command thread is a pixel command thread, such as thread 312, or a vertex command thread, such as thread 318. In this embodiment, the arbiter 306 selectively retrieves either a pixel command thread, such as command thread 316, or a vertex command thread, such as command thread 322.

[0026] ~~One~~ In one embodiment, each command thread within the reservation station 302 and 304 may be stored across two physical pieces of memory, wherein a majority of bits are stored in a ~~one report~~ one read port device. The bits required for the thread arbitration may be stored in a highly multi-ported structure, such that the bit stored in the one read port device are termed state bits and the bits stored in the multi-read port device are termed status bits.

[0027] In one embodiment the state bit includes, but is not limited to, a control flow instruction pointer, a loop iterater, a call return pointer, predicated bits, a GPR base pointer, a context pointer, valid bits, and any other suitable bits as recognized by one having skill in the art. It is also noted that in one embodiment, index pointers are not included in the state bits, wherein one embodiment may be stored in the general processing registers.

[0028] In this ~~embodiment the~~ embodiment, the fields of the state bits, the control flow ~~instructions pointers~~ instruction pointer, the execution count ~~marker loop~~ marker, loop iteraters, call return pointers, predicate bits, are updated every time the thread is returned to the reservation station 302 or 304 based on how much progress has been made on the thread execution. It is also noted that in this embodiments, the GPR base pointer and context pointers are unchanged throughout the execution of the thread.

[0029] In one ~~embodiment the~~ embodiment, the status bits ~~include a~~ include: a valid thread bit, a texture/ALU engine needed bit, a texture reads are outstanding bit and a waiting on texture read to complete bit. In this embodiment, all of the above status bit fields from the command threads ~~going~~ go to the arbitration circuitry. Thereupon, the arbiter 306 selects the proper allocation of which command thread goes to the graphics processing ~~agent~~ engine 310 ~~in~~ and which command thread goes to the ALU 308. In this embodiment, two sets of arbitration are ~~performed one~~ performed: one for pixels, such as command thread 316 and one for vertices, such as command thread 322. Although, texture arbitration requires no allocation or ordering as it is purely based on selecting the oldest thread that requires the graphics processing engine 310.

[0030] FIG. 5 illustrates a block diagram representing the further execution of the command threads upon completion of all embedded commands therein. The ALU 308 is coupled to a render backend 350 via connection 352 and to a scan converter 356 via connection ~~358~~ 354. As recognized by one having ordinary skill in the art, the ALU 308 may be operably coupled to the render backend 350 such that the bus 352 incorporates one or more of a plurality of connections for providing the completed command thread, such as command thread 316 of FIG. 4, thereto. Furthermore, as recognized by one having ordinary skill in the art, ALU 308 may be operably coupled to the scan converter 356 such that the connection ~~358~~ 354 may be one or more of a

plurality of connections for providing the executed command thread, such as command thread 322 of FIG. 4, to the scan converter 356. As discussed above, once the command ~~thread, have~~ thread's ~~an~~ indicator bit, such as the done flag, ~~set indicating~~ is set, indicating all of the commands in the thread have been executed, the completed command thread is further provided in the processing pipeline. Moreover, the render backend 350 may be any suitable rendering backend for graphics processing as recognized by one having ordinary skill in the art. The scan converter 356 may be any suitable scan converter for graphics processing as recognized by one having ordinary skill in the art.

[0032] The method further includes performing a command in response to the selected command thread, step 406. In this embodiment the command is performed by the graphics processing engine 310, which may be performing a texture operation. The next step, step 408, is writing the selected command thread to a first reservation station if the selected command thread is one of the plurality of first command threads and writing the selected command thread to a second reservation station if the selected command thread is one of the plurality of second command threads. With regard to FIG. 4, if the selected command thread is a pixel command thread, such as command thread 312-316, the graphics processing engine 310 provides the command thread 312-316 back thereto via connection ~~334~~ 332. Furthermore, if the command thread is from the vertex reservation station 304, the command thread 318-320 may be provided thereto via connection 334 from the graphics processing engine 310. Thereupon, the method is complete, step 410.

[0037] The next step, step 436, is writing the second selected command thread to a first reservation station if the selected command thread is one of a plurality of first command threads and writing the second selected command thread to a second reservation station if the second

selected command thread is one of a plurality of second command threads. Furthermore, the method includes writing the selected command thread to the first reservation station if the selected command thread is one of the plurality of first command threads and writing the selected command thread to the second reservation station if the selected command thread is one of the plurality of second command threads, step 438. Once again, using the exemplary embodiment of FIG. 4, the command threads, such as 312-316 and/or 318-322 may be provided from the graphics processing engine 310 and written back thereto or in another embodiment may be provided to the ALU 308 by the arbiter 306 and, upon execution of an arithmetic command, provided back to the associated reservation station, 302 or 304 respectively.